Compte Rendu de Projet – Processeur ARM7TDMI

Sanghyeon PARK | Florian LAINE

Informations :

- **SOFS** : Les sofs sont stockées dans le dossier /fit :
  - `fit/partie5.sof`
  - `fit/partie6.sof`
  - `fit/partie7.sof`
- **Pour les simulations/tests** :
  - `simu/compile.do` : qui compile tout les sources
  - `simu/test_benchs.do` : qui exécute l'ensemble de tout les tests.

1 Partie 1 - Unité de traitement

1.1 ALU

Entrées et sorties

- **Entrées** :
  - `OP` : Opcode sur 3 bits définissant l'opération à effectuer.
  - `A, B` : Opérandes.
  - `FLAGS` : Bits NZCV (Negative, Zero, Carry, Overflow).
- **Sortie** :
  - `S` : Résultat de l'opération.

Fonctionnement général

L'ALU exécute des opérations arithmétiques et logiques selon l'opcode (3 bits). Les opérations implémentées incluent ADD, SUB, AND, OR, XOR, NOT, ainsi que des opérations de transfert (A, B). Les flags NZCV sont mis à jour après chaque opération : - N (Negative) : 1 si résultat négatif - Z (Zero) : 1 si résultat nul - C (Carry) : 1 si retenue - V (Overflow) : 1 si débordement signé

1.2 Banc de registres

Entrées et sorties

- **Entrées** :
  - `CLK, RST, WE` : Respectivement Horloge, Reset et Write enable.
  - `W` : Bus d'ecriture sur 32 bits.
  - `RA, RB, RW` : Respectivement adresse a passer vers A, vers B et adresse qui va etre ecrite par le bus W.
- **Sortie** :
  - `A, B` : Sortie A et B

Fonctionnement général

Le banc gère 16 registres 32 bits. La lecture est combinatoire et simultanée sur les ports A/B. L'écriture est synchrone (front montant CLK) si WE=1. Le registre R15 est initialisé à 0x30. Une protection empêche l'accès à des indices invalides.

1.3 Mémoire de données

Entrées et sorties

- **Entrées** :
  - `CLK, RST, WE` : Respectivement Horloge, Reset et Write enable.
  - `Addr` : Adresse memoire a lire/ecrire
  - `DataIn` : Donnee a ecrire dans 'Addr' lors d'une ecriture
- **Sortie** :
  - `DataOut` : Les donnee lue a 'Addr' sont envoye par cette sortie.

Fonctionnement général

Mémoire 64x32 bits avec interface similaire au banc de registres. La lecture est combinatoire, l'écriture synchrone (WE=1). L'adresse utilise les 6 LSB du bus d'entrée.

1.4 Modules utilitaires

- **Multiplexeur 2→1** : Sélection entre A et B via COM
- **Extension de signe** : Étend un entier signé de N→32 bits

2 Partie 2 – Unité de gestion des instructions

Entrées et sorties

- **Entrées :**
  - `offset` : décalage sur 24 bits à ajouter au PC (type `pc_offset_t`).
  - `nPCsel` : sélectionne la mise à jour du PC.
  - `CLK` : horloge.
  - `RST` : reset asynchrone (actif à 1).
  - `IRQ` : signal d'interruption.
  - `IRQEnd` : fin de traitement de l'interruption.
  - `VICPC` : adresse du sous programme de l'interruption.
  - `FLAGS` : Flags avant l'interruption
- **Sorties :**
  - `instruction` : instruction 32 bits courante (type `word_t`).
  - `IRQServ` : acquittement de l'interruption.
  - `SAVED_FLAGS` : Signal qui sauvegarde les flags d'avant interruption pour ainsi les restaurer a la fin de l'interruption.

Fonctionnement général

Gère le PC et l'accès à la mémoire d'instructions (64x32 bits). Le PC est mis à jour selon nPCsel :

- nPCsel=0 : PC + 1
- nPCsel=1 : PC + 1 + offset étendu (24→32 bits)

3 Partie 3 - Unité de contrôle

3.1 Registre PSR

Entrées et sorties

- **Entrées** :
  - `CLK, RST, WE` : Respectivement Horloge, Reset et Write enable.
  - `DataIn` : Data a ecire si WE = 1
- **Sortie** :
  - `DataOut` : Data de sortie. Garde la meme valeur tant que WE = 0.

Fonctionnement général

Registre 32 bits stockant l'état du processeur (flags NZCV en bits 3-0 dans notre implémentation.) et une valeur d'affichage (RegAff). Chargement synchrone si WE=1.

3.2 Décodeur

Entrées et sorties

- **Entrées** :
  - `instruction` : L'instruction a parser.
  - `PSR` : Pour avoir l'etat du processeur a l'instant T. Le decodeur s'interessera au 4 bits de poids forts (les flags). Plus precisement le flag N pour l'instruction BLT.
- **Sortie** :
  - `nPCSel` : Si PC = 0, PC s'incremente de 1, donc suit l'execution normal du programme, sinon, Il va permettre de faire des branchement, notament avec BLT ou BAL par exemple.
  - `RegWr` : Si l'instruction doit ecrire dans un registre
  - `ALUSrc` : Determine si B viens du banc de registre ou est une valeur immediate
  - `ALUCtr` : L'opcode passe a l'alu
  - `PSREn` : Si on utilise les flags stocke par le PSR ;
  - `MemWr` : Si 'instruction ecrit dans la memoire de donnees
  - `WrSrc` : Determine si le bus W viens de l'alu ou de la memoire de donnees ;
  - `RegSel` : Force l'utilisation de Rd pour Rb
  - `RegAff` : Si il faut afficher dans les 4 seven segments
  - `IRQEnd` : Detecte une fin d'interruption.
  - `A, B` : Sortie A et B

Fonctionnement général

Génère les signaux de contrôle à partir de l'instruction et des flags. Deux processus : 1. Détermine le type d'instruction (MOV, ADDi, STR...) 2. Génère les commandes (ALUSrc, MemWr...) Le tableau des commandes est fourni page 9.

Tableau

Instr	nPCSel	RegWr	AluSrc	AluCtr	PsrEn	MemWr	WrSrc	RegSel	RegAff	IrqEnd
ADDi	0	1	1	000	0	0	0	0	0	0
ADDr	0	1	0	000	0	0	0	0	0	0
BAL	1	0	0	000	0	0	0	0	0	0
BEQ	1	0	0	000	0	0	0	0	0	0
BLT	1	0	0	000	0	0	0	0	0	0
BX	0	0	0	000	0	0	0	0	0	1
CMP	0	0	1	010	1	0	0	0	0	0
LDR	0	1	1	000	0	0	1	1	0	0
MOV	0	1	1	001	0	0	0	0	0	0
STR	0	0	1	000	0	1	0	1	1	0
NOP	0	0	0	000	0	0	0	0	0	0

Importants Les conditions sont considérées valides : Par exemple pour BEQ, nPCSel = 1 si Z = 1. et pour BLT nPCSel = 1 si N = 1

4 Partie 4 - Assemblage et validation du processeur

L'ensemble du processeur est intégré dans `Processor.vhd`. Il connecte tous les sous-systèmes entre eux.

Le test affiche `000A` (hexadécimal) sur les afficheurs 7 segments. Les adresses `0x20` à `0x2A` contiennent des 1 ; leur somme donne bien `0xA`.

5 Partie 5 - Test sur carte FPGA

Le projet FPGA est situé dans le dossier `fit` de l'archive fournie.

6 Partie 6 - Gestion des interruptions

6.1 VIC

Entrées et sorties

- **Entrées** :
  - `CLK, RST` : Respectivement l'horloge et le reset.
  - `IRQServ` : Signal d'acquittement de l'interruption.
  - `IRQ0, IRQ1` : Signals d'interruption.
- **Sortie** :
  - `IRQ` : Signal d'acquittement de l'interruption.
  - `VICPC` : addresse du sous programme de l'interruption

Fonctionnement général

Gère deux interruptions prioritaires (IRQ0 > IRQ1) :
- Détection front montant sur IRQ0/IRQ1

- Génère IRQ et adresse du handler (VICPC=0x9 ou 0x15)
- Attend IRQ_SERV pour réinitialiser la demande

Les flags sont sauvegardées pour pouvoir les restaurer après la fin de l'interruption (voir pc_manager.vhd).

6.2 Modifications

— **Décodeur** : Ajout de BX (fin d'interruption) et IRQ_END
— **Gestion des instructions** : Sauvegarde PC dans LR lors d'IRQ, restauration PC ← LR+1 sur IRQ_END

7 Partie 7 - Interface UART du processeur

7.1 UART_DEVICE

Entrées et sorties

— **Entrées** :
  — `CLK, RST` : Respectivement Horloge, Reset
  — `UART_Conf` : Octet à transmettre (8 bits)
  — `GPIO(1)` : Réception UART (Rx)
  — `GO` : Lance la transmission (actif haut)
— **Sorties** :
  — `GPIO(0)` : Transmission UART (Tx)
  — `DATA_RCV` : Octet reçu
  — `DATA_AVAIL` : Donnée reçue disponible (actif haut)
  — `TXIRQ` : Interruption en fin de transmission

Fonctionnement général

L'UART gère la communication série full-duplex avec :
— **Émission (TX)** :
  — Transmission lancée par `GO=1` si le module est libre (`TxBusy=0`)
  — Génère une interruption (`TXIRQ`) après l'envoi complet d'un octet
  — Utilise un diviseur de fréquence (`FDIV`) pour le baud rate
— **Réception (RX)** :
  — Détection automatique de trame sur `GPIO(1)`
  — Signal `DATA_AVAIL` mis à 1 quand un octet est reçu
  — Synchronisation via diviseur de fréquence dédié
— **Intégration processeur** :
  — Adressage mémoire : Accès via `STR 0x40` pour écrire `UART_Conf`
  — Gestion d'interruption : `TXIRQ` peut déclencher une IRQ via le VIC